

**ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN**

UNIVERSIDAD DE CANTABRIA



Trabajo Fin de Grado

**Diseño y validación de sistemas embebidos
para la monitorización y análisis de modos
de conducción**

**(Design and validation of embedded systems
for monitoring and analysis of driving modes)**

Para acceder al Título de

**Graduado en
Ingeniería de Tecnologías de Telecomunicación**

Autor: Jose Luis Lloreda Aresti

Septiembre - 2020

RESUMEN

El internet de las cosas está teniendo cada vez más importancia y abarcando más ámbitos de nuestra cotidianidad, desde una simple bombilla conectada a la red Wi-Fi de casa, hasta la monitorización de pacientes en los hospitales. El vehículo autónomo es otro interesante ejemplo de aplicación, siendo el propósito final la eliminación de cualquier tipo de intervención humana. Con estas condiciones de contorno, el objetivo de este trabajo es el diseño de un dispositivo de bajo coste y fácil uso que sea capaz de detectar distintos modos de conducción. Ello debe permitir mejorar la manera de entender las formas de conducir de las personas para así ayudar a mejorar los flujos de tráfico, así como disminuir el número de accidentes a la vez que se facilita la sostenibilidad.

ABSTRACT

The Internet of Things is becoming a key topic spanning more and more verticals. From a simple light bulb connected to the home Wi-Fi network, to patient monitoring in hospitals. The autonomous vehicle is another example of the IoT in our everyday live, where no human interaction with the vehicle is the end goal. It is in this framework that the present document aims at designing and assessing a low-cost and easy-to-use device that can detect different driving modes. Which can help to understand better the way people drives, in order to improve the traffic flows while reducing the number of accidents and increasing the sustainability.

Índice general

Índice general	1
Índice de figuras	2
GLOSARIO	3
1. Introducción	4
1.1 Objetivos	4
1.2 Estructura	5
2. Descripción de la problemática e identificación de las técnicas y herramientas de análisis y caracterización	6
2.1 Requisitos técnicos	6
3. Desarrollo hardware/software	10
3.1. Subsistema hardware	10
3.2. Subsistema software	13
4. Caracterización de prestaciones	19
4.1 Caracterización del giróscopo	21
4.2 Caracterización de acelerómetro	22
5. Conclusiones y futuras líneas de trabajo	26
Referencias bibliográficas	27

Índice de figuras

Figura 2.1: Ejes de referencia de acelerómetro y giroscopio.....	8
Figura 2.2: Ejes de referencia de magnetómetro.....	8
Figura 3.1: Microcontrolador Arduino Fio.....	11
Figura 3.2: Integrado MPU-9250 IMU.	11
Figura 3.3: Módulo transmisor Xbee.	12
Figura 3.4: Módulo transmisor Xbee integrado en el microcontrolador Arduino Fio. ..	12
Figura 3.5: Módulo receptor Xbee.	13
Figura 3.6: Interfaz gráfica del IDE Arduino, donde se identifica las dos funciones principales.....	13
Figura 3.7: Ejemplo de cómo incluir una librería.	14
Figura 3.8: Ejemplo de cómo declarar distintas variables.....	14
Figura 3.9: Ejemplo de una parte de la función void setup()utilizada en este trabajo. .	15
Figura 3.10: Mensaje de conexión fallida.....	15
Figura 3.11: Instrucciones de la calibración inicial del equipo.....	16
Figura 3.12: Código relativo a la toma de datos de los 9 ejes del MPU-6050 IMU.....	17
Figura 3.13: Fusión de los datos en una sola cadena y su envío.	18
Figura 4.1: Detalle del equipo de medida embarcado en el tren.	19
Figura 4.2: Detalle del trazado de medida inicial.	20
Figura 4.3: Medidas en los ejes X e Y de proporcionadas por el acelerómetro.....	20
Figura 4.4: Detalle del trazado circular.....	21
Figura 4.5: Medidas proporcionadas por el giróscopo.....	21
Figura 4.6: Medida del valor de la componente z del giróscopo.	22
Figura 4.7 Detalle del trazado para la caracterización del acelerómetro.	23
Figura 4.8: Medida del módulo de las componentes A_x y A_y	24
Figura 4.9: Medidas del acelerómetro teniendo en cuenta los signos.	24
Figura 4.10 Gráfica del trayecto en coche del integrado junto con GPS.	25

GLOSARIO

EEPROM Electrically Erasable Programmable Read-Only Memory.

GPS Global Positioning System.

GUI Graphical User Interface.

I2C Inter Integrated Circuits.

IDE Integrated Development Environment.

IMU inertial Measurement Unit.

IoT Internet of Things.

PWM Pulse Width Modulation.

RFID Radio Frequency Identification.

SRAM Static Random Access Memory.

SPI Serial Peripheral Interface.

Capítulo 1

Introducción

En la actualidad la Internet de las cosas está presente en una gran variedad de formas en nuestras vidas, desde las ciudades inteligentes hasta la industria 4.0, pero no es tan reciente como pueda parecer pues se conoce que el primer dispositivo conectado a internet fue una máquina de vending, en la Universidad de Carnegie Mellon, en Pittsburgh, Estados Unidos, conectada a la red ARPANET en el año 1982. Aunque la primera vez que dicho termino aparece de forma explícita es en 1999, de la mano de Kevin Ashton, quien trabajaba en un proyecto relacionado con tecnología RFID [1], y tecnologías de sensores.

Las aplicaciones de la Internet de las cosas son muy variadas, abarcando el ámbito domótico haciendo realidad el paradigma de edificios y viviendas sostenibles en los que, a modo de ejemplo, los diferentes parámetros relativos a la calidad del aire, consumo o temperatura son monitorizados de forma continua a fin de proporcionar las mejores condiciones para sus inquilinos a la vez que se optimiza el consumo de energías. Pero también cabe pensar en la posibilidad de monitorizar y teleactuar sobre diferentes electrodomésticos o equipos en base a las necesidades concretas del día a día de los residentes. Más exigentes son los requerimientos que emanan de la aplicación de IoT al ámbito de la salud haciendo posible una mejor gestión de los pacientes hospitalizados, pero a la vez reduciendo las estancias de estos en los hospitales enviándolos a sus lugares de residencia habitual dotándoles de dispositivos móviles e itinerantes que facilitan el acceso a los parámetros precisos en función de las necesidades del proceso de recuperación. Entre aquellos requerimientos cabe destacar los relativos a la privacidad y seguridad de la información, pero también la propia resiliencia de los dispositivos y la arquitectura bajo la que operan.

1.1 Objetivos

Uno de los ámbitos más complejos y a la vez más interesantes y con mayor proyección de futuro está en el transporte donde el vehículo autónomo [2] sin necesidad de interacción humana, (vehículo autónomo de nivel 5), es el objetivo último. Soluciones

con mínima intervención humana deberían tener importantes repercusiones a la hora de reducir el número de accidentes a la vez que se mejoran las condiciones del tráfico con las consecuentes implicaciones en términos de reducción de consumos o en la duración de los trayectos. Aunque son muchos los progresos que se están realizando en este ámbito no es menos cierto que todavía queda un largo camino por recorrer. Sin embargo, hasta alcanzar ese objetivo de plena autonomía hay varias fases intermedias que ayudan a comprender, entre otros, los hábitos de conducción lo que en iteraciones posteriores debe permitir optimizar los diseños al efecto. Muy particularmente, en el transporte de mercancías por carretera la caracterización de determinados patrones y la eventual notificación de alarmas está llamada a jugar un papel relevante para lo que se deben diseñar, implementar y validar soluciones basadas en equipos relativamente económicos de fácil uso. Alineado con esta tendencia el presente trabajo aborda el diseño y validación de un prototipo que permita la detección de aceleraciones y giros bruscos, entendiéndose estos como aquellos que superan un determinado umbral a fijar por el gestor de la flota de vehículos.

1.2 Estructura

El presente trabajo, una vez proporcionada la correspondiente introducción y fijados los objetivos incluye un Capítulo 2 dedicado a describir la problemática general que se aborda, así como la captura de requisitos que debe satisfacer el correspondiente diseño. El Capítulo 3 muestra los diferentes componentes usados, así como su software, mientras que el Capítulo 4 se dedica a la caracterización de las prestaciones del equipo con las diferentes configuraciones que se han usado. Y por último el Capítulo 5 plantea las conclusiones y las líneas futuras de trabajo.

Capítulo 2

Descripción de la problemática e identificación de las técnicas y herramientas de análisis y caracterización

Tal como se ha adelantado, el propósito de este trabajo es diseñar un prototipo de equipo de bajo coste y con necesidades de calibración reducidas que posibilite monitorizar la conducción de diferentes tipos de vehículos, en especial camiones, con el objetivo de analizar las características más destacables que deban ser objeto de mejora, por parte de cada conductor, a fin de gestionar de forma sostenible las diferentes flotas de vehículos. Aspectos relativos a aceleraciones y deceleraciones extremadamente bruscas, o velocidades centrífugas con valores próximos a los límites son algunos de los parámetros que se buscan identificar.

En el mercado existen soluciones que tienen varios inconvenientes. Entre ellos, el más frecuente es la exigencia de los equipos en las tareas de calibración que además de ser enormemente extensas desde un punto de vista temporal son también complejas. Ello hace que los conductores en numerosas ocasiones decidan no activar el equipo amparándose en la complejidad de los procesos citados. Y por supuesto, además de los aspectos técnicos también hay que tener en cuenta el elevado coste de muchos de los equipos, máxime cuando se tienen que replicar a nivel de cada uno de los vehículos de la flota a gestionar.

2.1 Requisitos técnicos

Se recogen seguidamente los requerimientos técnicos básicos que la solución a implementar debe verificar:

- 1) Detección de aceleraciones lineales.
- 2) Detección de deceleraciones lineales.
- 3) Operación con 6 y/o 9 ejes que se corresponden con acelerómetro, giróscopo y magnetómetro.

4) Posibilidad de emplear GPS con objeto de incrementar la precisión de determinadas medidas.

5) Utilización de dispositivos embebidos de amplia difusión.

Para cumplir con los citados requerimientos, inicialmente se postuló utilizar el circuito integrado MPU-6050 (seis ejes) [3] junto con la placa de Arduino Uno [4], elección que se descartó en favor de la tarjeta MPU-9250 dotada de los nueve ejes [5] junto la placa Arduino fío [6].

Es importante saber qué son, cómo funcionan y qué miden cada uno de los nueve ejes de los que dispone la tarjeta MPU-9250, lo cual se muestra a continuación.

2.1.1 Acerca del giróscopo

Un giróscopo o giroscopio es un equipo que mide el movimiento de rotación, respecto a un eje, del cuerpo en el que está acoplado. Mide cambios de posición angular, generalmente en grados o radianes por segundo. La integración de esta información a lo largo del tiempo permite identificar los cambios de orientación experimentados. Estos sensores disponen de tres ejes perpendiculares entre sí, realizando la captura de información de las rotaciones que sufre el cuerpo. Como resultado, se obtienen las velocidades angulares en cada uno de sus tres ejes, es decir, la velocidad a la que el cuerpo rota en cada uno de dichos ejes, como se puede apreciar en la figura 2.1.

2.1.2 Acerca del acelerómetro

Un acelerómetro es un equipo capaz de cuantificar la aceleración que experimenta un cuerpo en movimiento. Mide aceleraciones lineales. Para el proyecto se hablará nuevamente de acelerómetros digitales/ electrónicos, que al igual que los giróscopos cuentan con tres ejes. Consecuentemente tres serán las medidas que arrojen, correspondiéndose a la aceleración sufrida en cada uno de ellos, como se muestra en la figura 2.1.

2.1.3 Acerca del magnetómetro

Un magnetómetro es un dispositivo que mide la fuerza y dirección de un campo magnético. En navegación, aprovechando el conocido campo magnético terrestre se emplea para identificar la dirección Norte y este, siendo capaz de ubicar un punto en el espacio respecto de un sistema prefijado referenciado al campo magnético terrestre. Es necesario señalar, que los magnetómetros son susceptibles a las distorsiones externas sobre el campo magnético terrestre, lo que modificará los valores detectados e incurrirá en errores. Esta circunstancia, conocida como distorsiones de hierro blando y hierro duro deberán tenerse en cuenta y realizar los correspondientes cálculos

compensatorios. Al contrario que los anteriores muestra una disposición de los ejes distinta, como se puede ver en la figura 2.2. Si bien en los casos previos relativos al acelerómetro y giroscopio el dato está claramente expresado respecto de los ejes propios, en este, la ubicación tanto del centro como del primer eje está condicionada por la posición relativa del equipo a medida que se mueve sobre la superficie de la tierra. Aprovechando el campo magnético terrestre conocido, el magnetómetro será el equipo cuyo eje “X” en todo momento apunte al norte magnético de la Tierra. A partir del anterior, se considera eje “Y” a aquel que apunta al este. El eje “Z” será perpendicular al plano formado por los dos anteriores y su sentido positivo hacia abajo (no tiene por qué pasar por el centro de la Tierra).

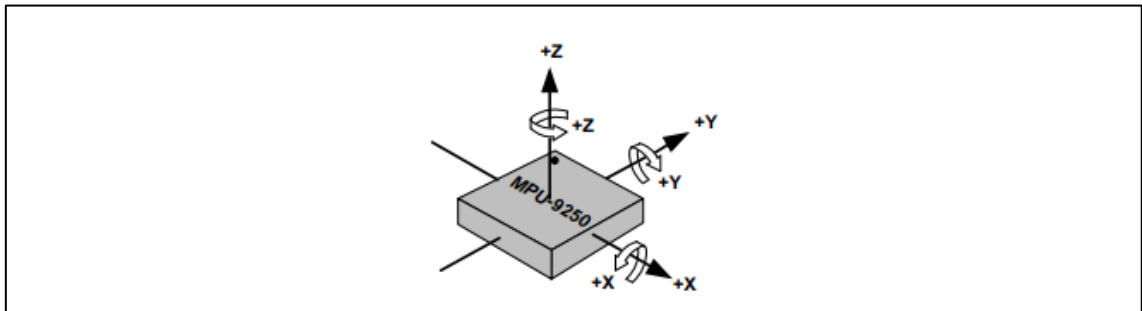


Figura 2.1: Ejes de referencia de acelerómetro y giroscopio.

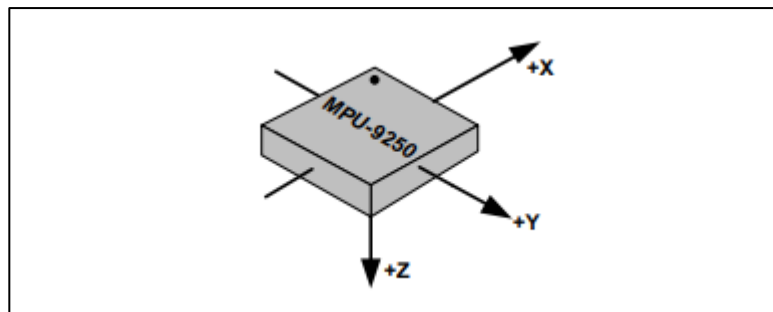


Figura 2.2: Ejes de referencia de magnetómetro.

A partir de los datos brutos que otorgan el giroscopio, acelerómetro, magnetómetro y, eventualmente, el GPS se deberá obtener la trayectoria completa del vehículo en el que están instalados. Como consecuencia de esos 6, 9 ó 10 datos que arrojan los sensores se debe obtener la posición y la rotación experimentada en cada instante de tiempo. Es necesario puntualizar que la información que aportan estos equipos es una posición relativa respecto a un punto de partida conocido.

Para poder realizar esta transformación y puesta en claro de la información es necesario invocar al paradigma de la fusión de datos, mediante la cual se consigue

mejorar la precisión de estos a través de la combinación apropiada de los citados datos proporcionados por el acelerómetro y el giroscopio principalmente.

El principal motivo del porqué esta integración de datos es necesaria reposa en la deriva y ruido que los equipos incluyen en los datos reportados.

De entre las técnicas empleadas para la fusión de los datos cabe destacar el del filtro complementario [7], el cual, desde un punto de vista computacional es el menos costoso. Esencialmente, consiste en aplicar un “filtro de paso bajo” al acelerómetro, es decir, dejar pasar los valores que se encuentran por debajo de un cierto límite. Por el contrario, para los datos reportados por el giroscopio se utilizará un “filtro de paso alto”, utilizando por lo tanto sólo aquellos que se encuentran por encima de cierto valor.

Estos métodos no son ni mucho menos ideales puesto que conllevan errores de diferente naturaleza, pero teniendo en cuenta que requieren de una menor capacidad de cómputo y el objetivo principal es el diseño de un equipo de bajo coste, lo que implica emplear sistemas limitada capacidad de cómputo, son errores perfectamente asumibles.

Como paso previo a la realización de estos cálculos ligado a cada uno de los tres dispositivos, giróscopo, acelerómetro y magnetómetro, es preciso el cálculo de la matriz de rotación [8]. Para el cálculo de dicha matriz se pueden usar diferentes métodos o modelos como son la matriz de cosenos directores [9], ángulos de Euler [10] o los cuaterniones [11].

Estos son en esencia una extensión de los números complejos. Si los números complejos se caracterizan por añadir una componente imaginaria, “ i ”, los cuaterniones aportan además la “ j ” y la “ k ”. Consecuentemente un cuaternión tendrá parte real y las tres imaginarias i , j , k . La ventaja del uso de cuaterniones reside en que su interpretación se asemeja a la física de los métodos anteriormente nombrados, facilitando su comprensión, resultando de especial utilidad para representar rotaciones de un cuerpo respecto de otro. Por el contrario, no son vectores y algunas propiedades vectoriales no se cumplen rigiéndose las operaciones con cuaterniones por reglas propias. Es de especial relevancia el hecho de que el producto entre cuaterniones siendo asociativo, no es conmutativo. Esta consideración deberá tenerse en cuenta en las interpretaciones de resultados, si bien actualmente estas operaciones están resueltas en diferentes entornos de programación por lo que apenas resultará limitante.

Capítulo 3

Desarrollo hardware/software

Se presentan aquí los diferentes subsistemas que componen el hardware, así como software utilizado por cada uno de ellos. Dada la necesidad de disponer de una solución basada en una plataforma de amplia difusión el hardware elegido emana del entorno de Arduino el cual presenta una amplia comunidad de desarrolladores lo que facilita solventar determinadas problemáticas que se pueden dar en la fase de desarrollo.

3.1. Subsistema hardware

3.1.1 Microprocesador embebido

Aunque inicialmente se había planteado emplear el Arduino Uno junto al MPU-6050 IMU, finalmente el equipo utilizado está formado por, el microcontrolador Arduino Fio, el integrado MPU-9250 IMU y los dispositivos radio Xbee (transmisor y receptor) [12].

La base del equipo es el microcontrolador Arduino Fio, el cual trabaja a una tensión de alimentación de 3'3V y a una frecuencia de reloj de 8 MHz. Cuenta con un total de 14 entradas/salidas digitales, 6 de ellas se pueden utilizar como salidas PWM, y 8 entradas analógicas. Dispone de una memoria flash de 32 KB, una SRAM de 2 KB y una EEPROM de 1 KB. También tiene la capacidad de poder conectar baterías de polímero de litio y lo más importante y motivo de su elección, cuenta con un adaptador de Xbee integrado, como se observa en la figura 3.1.

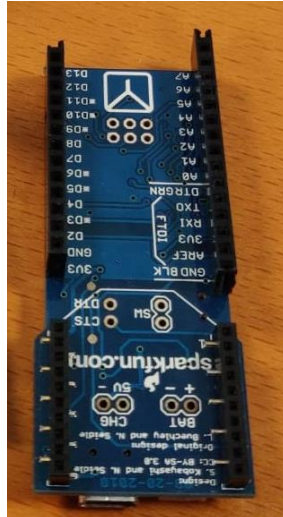


Figura 3.1: Microcontrolador Arduino Fio.

3.1.2 Circuito integrado IMU

Por otra parte, todos los datos son medidos desde el integrado MPU-9250 IMU, como el que se muestra en la figura 3.1.2.1 Este dispositivo cuenta, como ya se ha explicado anteriormente con un acelerómetro, un giróscopo y un magnetómetro. Las comunicaciones desde el integrados resultan sencillas ya que cuenta con un bus SPI, así como con un bus I2C. Opera con una tensión de entre 2'4V y 3'6V. Se puede afirmar que es un equipo de bajo consumo, pues con todos los sensores activados consumen solamente 3'5mA.

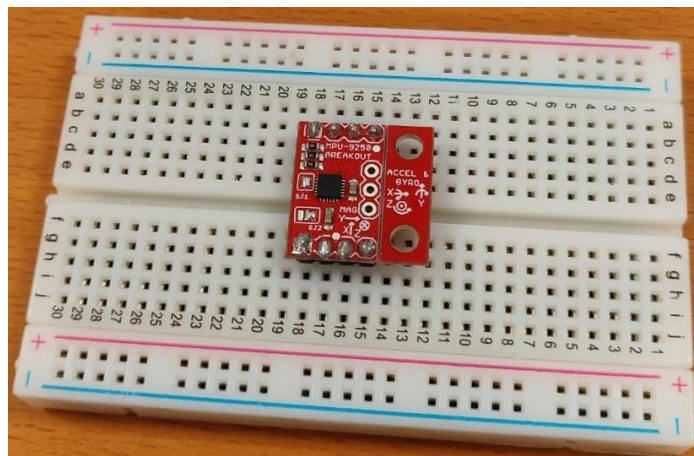


Figura 3.2: Integrado MPU-9250 IMU.

3.1.3 Dispositivos radio

Por último, el equipo cuenta con dos transceptores radio, los cuales se comunican de forma inalámbrica, los módulos Xbee. El principal uso de Xbee es sustituir una comunicación serie cableada por otra inalámbrica, mediante redes puntos a punto, punto a multipunto o redes malladas. Estas radios cuentan con varias entradas y salidas tanto analógicas como digitales. Una de sus grandes ventajas es su bajo consumo. Se basan en el estándar IEEE 802.15.4, el cual especifica los niveles físicos y de acceso al medio en redes inalámbricas con tasas de transmisión de datos bajas. Cabe decir que el citado estándar es la especificación de referencia para la solución inalámbrica conocida como ZigBee. Un detalle de los diferentes módulos indicados se muestra en las figuras 3.3, 3.4 y 3.5.



Figura 3.3: Módulo transmisor Xbee.

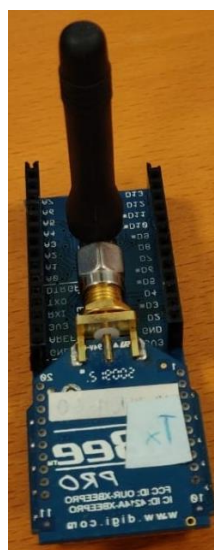


Figura 3.4: Módulo transmisor Xbee integrado en el microcontrolador Arduino Fio.

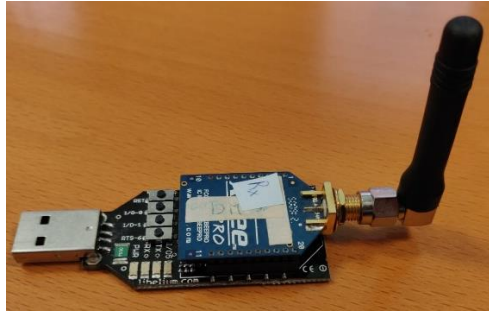


Figura 3.5: Módulo receptor Xbee.

3.2. Subsistema software

El software utilizado en el equipo es el de su microcontrolador el Arduino Fio, que emplea un entorno de desarrollo integrado Arduino IDE. El IDE de Arduino cuenta tanto con editor de código, como de un compilador y depurador interno. También posee una amigable interfaz gráfica de usuario, GUI, desde la cual se accede a la gestión de las librerías usadas y de la placa que se esté utilizando en el equipo.

Como todos los códigos Arduino el de este proyecto cuenta con dos partes claramente diferenciadas. A saber, las funciones principales *void setup()* y *void loop()*, tal como se muestra en la figura 3.6.

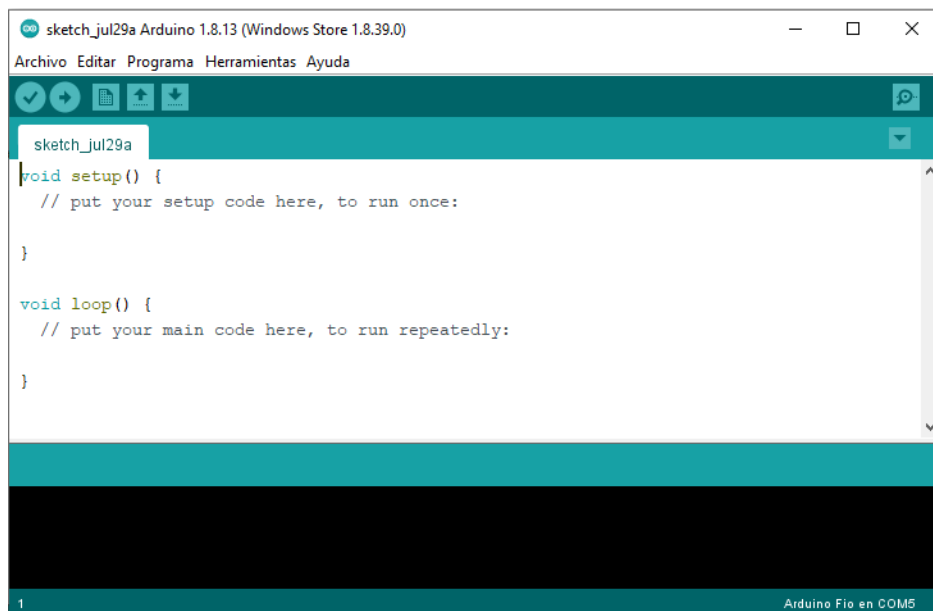


Figura 3.6: Interfaz gráfica del IDE Arduino, donde se identifica las dos funciones principales.

Todo código cuenta también con una parte opcional, siendo lo primero que se muestra debiéndose declarar en él aquellas librerías empleadas haciendo uso del

comando `#include` “nombre de la librería”. Así mismo, es donde se declaran todas las variables globales y eventualmente se inicializan. En las figuras 3.7 y 3.8 se muestra un extracto del código usado en este trabajo a modo de ejemplo.

```
#include "src/beemod/Bee.h"

//#include <avr/dtostrf.h>

#include "quaternionFilters.h"
#include "MPU9250.h"
#ifdef LCD
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
```

Figura 3.7: Ejemplo de cómo incluir una librería.

```
int count=0;
Bee bee = Bee(&Serial, 9600);
unsigned long timenow;
String count_text;

float Ax = 0;
float Ay = 0;
float Az = 0;
float Gx = 0;
float Gy = 0;
float Gz = 0;
float Mx = 0;
float My = 0;
float Mz = 0;
unsigned int i = 1;
```

Figura 3.8: Ejemplo de cómo declarar distintas variables.

La primera función principal se llama `void setup()`, en la que se establece la configuración necesaria para que el equipo opere de forma correcta. Estas instrucciones, al igual que las anteriormente descritas, se ejecutan solamente una vez al iniciar al equipo. Con ellas se caracterizan los pines de entrada/salida de los datos, la velocidad a la que trabaja la comunicación serie, la activación de los sensores necesarios del equipo para la recolección de los datos, entre otras, tal como se puede ver en la figura 3.9.

```

void setup() {
    // put your setup code here, to run once:
    bee.setCallback(beeCallback);
    bee.begin();

    Wire.begin();
    // TWBR = 12; // 400 kbit/sec I2C speed
    Serial.begin(9600);

    while(!Serial){};

    // Set up the interrupt pin, its set as active high, push-pull
    pinMode(intPin, INPUT);
    digitalWrite(intPin, LOW);
    pinMode(myLed, OUTPUT);
    digitalWrite(myLed, HIGH);

    // Read the WHO_AM_I register, this is a good test of communication
    byte c = myIMU.readByte(MPU9250_ADDRESS, WHO_AM_I_MPU9250);
    Serial.print(F("MPU9250 I AM 0x"));
    Serial.print(c, HEX);
    Serial.print(F(" I should be 0x"));
    Serial.println(0x71, HEX);
}

```

Figura 3.9: Ejemplo de una parte de la función void setup() utilizada en este trabajo.

Adicionalmente, en este proyecto también se ha utilizado la función *void setup()* para comprobar que el equipo ha arrancado de manera correcta, en caso negativo informa por pantalla que no se ha podido conectar y aborta la conexión, y en caso afirmativo se ejecutan una serie de instrucciones para realizar una calibración inicial, tal como se refleja en las figuras 3.10 y 3.11.

```

    } // if (c == 0x71)
    else
    {
        Serial.print("Could not connect to MPU9250: 0x");
        Serial.println(c, HEX);

        // Communication failed, stop here
        Serial.println(F("Communication failed, abort!"));
        Serial.flush();
        abort();
    }
}

```

Figura 3.10: Mensaje de conexión fallida.

```

if (c == 0x71) // WHO_AM_I should always be 0x71
{
    Serial.println(F("MPU9250 is online..."));

    // Start by performing self test and reporting values
    myIMU.MPU9250SelfTest(myIMU.selfTest);
    Serial.print(F("x-axis self test: acceleration trim within : "));
    Serial.print(myIMU.selfTest[0],1); Serial.println("% of factory value");
    Serial.print(F("y-axis self test: acceleration trim within : "));
    Serial.print(myIMU.selfTest[1],1); Serial.println("% of factory value");
    Serial.print(F("z-axis self test: acceleration trim within : "));
    Serial.print(myIMU.selfTest[2],1); Serial.println("% of factory value");
    Serial.print(F("x-axis self test: gyration trim within : "));
    Serial.print(myIMU.selfTest[3],1); Serial.println("% of factory value");
    Serial.print(F("y-axis self test: gyration trim within : "));
    Serial.print(myIMU.selfTest[4],1); Serial.println("% of factory value");
    Serial.print(F("z-axis self test: gyration trim within : "));
    Serial.print(myIMU.selfTest[5],1); Serial.println("% of factory value");

    // Calibrate gyro and accelerometers, load biases in bias registers
    myIMU.calibrateMPU9250(myIMU.gyroBias, myIMU.accelBias);

    myIMU.initMPU9250();
    // Initialize device for active mode read of accelerometer, gyroscope, and
    // temperature
    Serial.println("MPU9250 initialized for active data mode....");

    // Read the WHO_AM_I register of the magnetometer, this is a good test of
    // communication
    byte d = myIMU.readByte(AK8963_ADDRESS, WHO_AM_I_AK8963);
    Serial.print("AK8963 ");
    Serial.print("I AM 0x");
    Serial.print(d, HEX);
    Serial.print(" I should be 0x");
    Serial.println(0x48, HEX);
}

```

Figura 3.11: Instrucciones de la calibración inicial del equipo.

Por último está la segunda función principal, *void loop()*, siempre ha de estar declarada después de la *void setup()*. En esta parte del código es donde se escriben todas funciones, llamadas a funciones, bucles que el equipo necesita repetir de manera constante, para su correcto funcionamiento. En el caso concreto del código utilizado en este proyecto cabe destacar que, como se comentará más adelante en el Capítulo 4. relativo a la caracterización de prestaciones, se hizo frente a una serie de problemas relacionados con la calibración del equipo que se solucionaron vía código. Una vez realizada la calibración del equipo se procede a tomar las medidas de los 9 ejes con los que cuenta el integrado, como se muestra en la figura 3.12.

```

// If intPin goes high, all data registers have new data
// On interrupt, check if data ready interrupt
if (myIMU.readByte(MPU9250_ADDRESS, INT_STATUS) & 0x01)
{
    myIMU.readAccelData(myIMU.accelCount); // Read the x/y/z adc values

    // Now we'll calculate the acceleration value into actual g's
    // This depends on scale being set
    myIMU.ax = (float)myIMU.accelCount[0] * myIMU.aRes; // - myIMU.accelBias[0];
    myIMU.ay = (float)myIMU.accelCount[1] * myIMU.aRes; // - myIMU.accelBias[1];
    myIMU.az = (float)myIMU.accelCount[2] * myIMU.aRes; // - myIMU.accelBias[2];

    myIMU.readGyroData(myIMU.gyroCount); // Read the x/y/z adc values

    // Calculate the gyro value into actual degrees per second
    // This depends on scale being set
    myIMU.gx = (float)myIMU.gyroCount[0] * myIMU.gRes;
    myIMU.gy = (float)myIMU.gyroCount[1] * myIMU.gRes;
    myIMU.gz = (float)myIMU.gyroCount[2] * myIMU.gRes;

    myIMU.readMagData(myIMU.magCount); // Read the x/y/z adc values

    // Calculate the magnetometer values in milliGauss
    // Include factory calibration per data sheet and user environmental
    // corrections
    // Get actual magnetometer value, this depends on scale being set
    myIMU.mx = (float)myIMU.magCount[0] * myIMU.mRes
        * myIMU.factoryMagCalibration[0] - myIMU.magBias[0];
    myIMU.my = (float)myIMU.magCount[1] * myIMU.mRes
        * myIMU.factoryMagCalibration[1] - myIMU.magBias[1];
    myIMU.mz = (float)myIMU.magCount[2] * myIMU.mRes
        * myIMU.factoryMagCalibration[2] - myIMU.magBias[2];
} // if (readByte(MPU9250_ADDRESS, INT_STATUS) & 0x01)

```

Figura 3.12: Código relativo a la toma de datos de los 9 ejes del MPU-6050 IMU.

A continuación, los datos se pasan por los distintos métodos de análisis mencionados en el apartado 2.1.3, y fusionan en una única cadena de datos, la cual no puede sobrepasar los 73 bytes de longitud (tamaño máximo permitido por ZigBee), para ser enviados inalámbricamente mediante el comando correspondiente, tal como se ilustra en la figura 3.13.

```

// Acelerometro en mg          giroscopo en grados por segundo          magnetometro en grados por segundo

dtostrf(1000*myIMU.ax - DesfaseAx, 4, 2, buff);
strcat(valueString, buff);
strcat(valueString, ";");

dtostrf(1000*myIMU.ay - DesfaseAy, 4, 2, buff);
strcat(valueString, buff);
strcat(valueString, ";");

dtostrf(1000*myIMU.az + DesfaseAz, 4, 2, buff);
strcat(valueString, buff);
strcat(valueString, ";");

dtostrf(myIMU.gx, 4, 2, buff);
strcat(valueString, buff);
strcat(valueString, ";");

dtostrf(myIMU.gy, 4, 2, buff);
strcat(valueString, buff);
strcat(valueString, ";");

dtostrf(myIMU.gz, 4, 2, buff);
strcat(valueString, buff);
strcat(valueString, ";");

dtostrf(myIMU.mx, 4, 2, buff);
strcat(valueString, buff);
strcat(valueString, ";");

dtostrf(myIMU.my, 4, 2, buff);
strcat(valueString, buff);
strcat(valueString, ";");

dtostrf(myIMU.mz, 4, 2, buff);
strcat(valueString, buff);
strcat(valueString, ";");

dtostrf(myIMU.temperature, 4, 2, buff);
strcat(valueString, buff);
strcat(valueString, ";");

dtostrf(Nseq, 1, 0, buff);
// ltoa(Nseq,buff,10);
strcat(valueString, buff);
strcat(valueString, ";");

bee.sendData(valueString); // El tamaño maximo del payload es de 73 bytes//

```

Figura 3.13: Fusión de los datos en una sola cadena y su envío.

Capítulo 4

Caracterización de prestaciones

En este capítulo se va a explicar las diferentes configuraciones que se han empleado y en base a las cuales se ha caracterizado el integrado MPU-9250 IMU.

Todos los datos que aquí se presentan han sido recopilados usando el tren a escala 1:22,5 de la casa LGB, [12] el cual se muestra en figura 4.1. La razón de emplear inicialmente este tipo de infraestructura se debe, por un lado, a la facilidad con la que se pueden realizar diversos tipos de medidas y muy particularmente porque al disponer de trayectorias muy determinadas la repetitividad de las medidas, con unos parámetros muy específicos (entre ellos el radio de curvatura), queda garantizada. Así mismo, hay que indicar que el análisis realizado con esta disposición de medida persigue la caracterización de 6 ejes (giróscopo y acelerómetro) de los 9 que dispone el integrado (los seis anteriores más el magnetómetro).

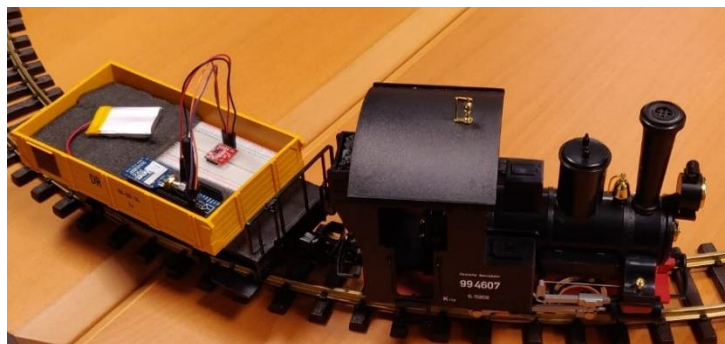


Figura 4.1: Detalle del equipo de medida embarcado en el tren.

La primera de las citadas configuraciones se basa en dos tramos rectos y otros tantos en forma de semicircunferencia (con radio de curvatura de 61 cm) tal como se muestra en la figura 4.2.

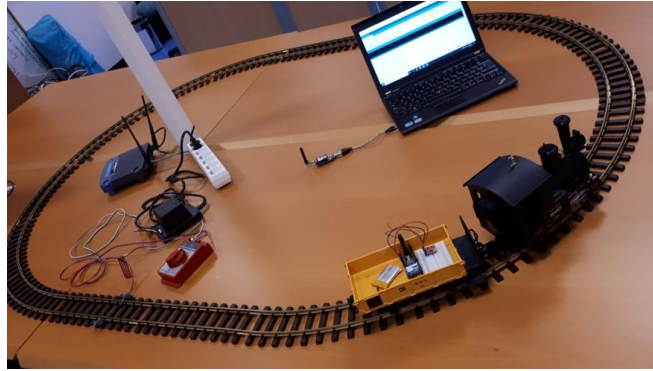


Figura 4.2: Detalle del trazado de medida inicial.

Tras varias medidas en este trazado lo primero que se advirtió fue la presencia de valores iniciales que, a priori, podían desvirtuar aquellas. En reposo, y con la componente peso perpendicular a la superficie de apoyo del mismo (componente Z) los valores teóricos que se deberían medir, en el acelerómetro son $A_x = 0 \text{ mG}$ (siendo A_x la componente X de la aceleración) $A_y = 0 \text{ mG}$ (con A_y la componente Y de la aceleración) y $A_z = 1000 \text{ mG}$ (con A_z la componente Z de la aceleración); donde $1 \text{ G} = 9.8 \text{ m/s}^2$ (a saber, la aceleración debida a la fuerza de la gravedad). Así mismo los datos del giróscopo, expresados en grados sexagesimales por segundo (esencialmente, la velocidad angular en torno a en cada uno de los ejes del cuerpo rotando) deberían ser $G_x = 0$, $G_y = 0$, $G_z = 0$. Pues bien, como se ha indicado, habitualmente en el momento de arrancar el circuito de medida junto con la plataforma de Arduino los datos del acelerómetro medidos presentaban sesgos, que en algunos casos, alcanzaban casi los 100 mG. En la figura 4.3 se muestra un ejemplo al respecto de lo comentado y que será objeto de análisis un poco más adelante.

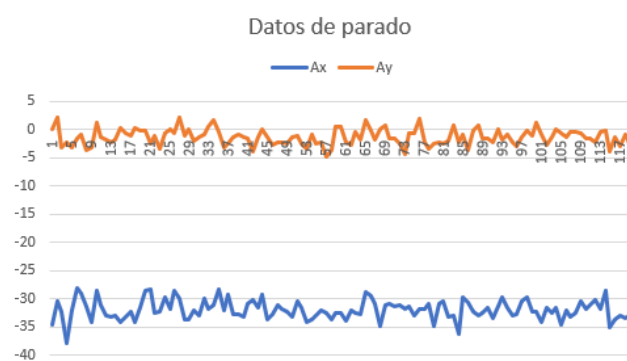


Figura 4.3: Medidas en los ejes X e Y de proporcionadas por el acelerómetro.

4.1 Caracterización del giróscopo

Para la caracterización del giróscopo, y a fin de hacerla de la forma más exacta posible se quitaron del trazado del tren los tramos rectos para así disponer de un trazado puramente circunferencial, como se puede observar en la figura 4.4, lo que permitiría disponer de una velocidad angular constante sin más que alimentar la locomotora con una tensión fija.

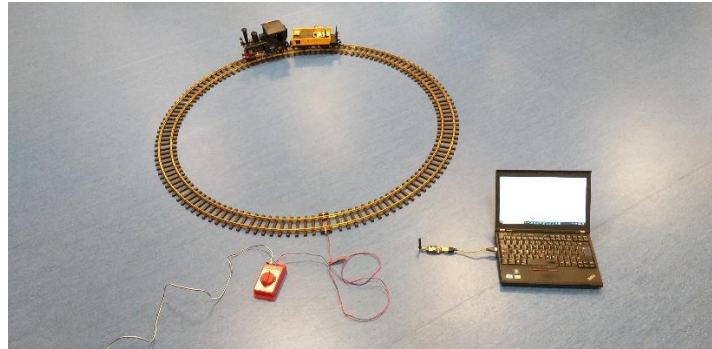


Figura 4.4: Detalle del trazado circular.

Observar que, a priori, las medidas para los ejes X e Y, deberían salir 0 y únicamente ser distinta de cero la componente A_z . En el caso que nos ocupa, para el radio de curvatura dado, la locomotora realiza una vuelta completa en 25 s (lo que equivale a 50 muestras dado que se está muestreando la salida del giróscopo a razón de 2 muestras/s), esto es, $0,25 \text{ rad/s}$ (lo que equivale a $14,5^\circ/\text{s}$). En la figura 4.5 se muestran los valores recogidos en módulo (para cada una de las muestras) verificándose la gran concordancia entre los valores medidos y el teórico.

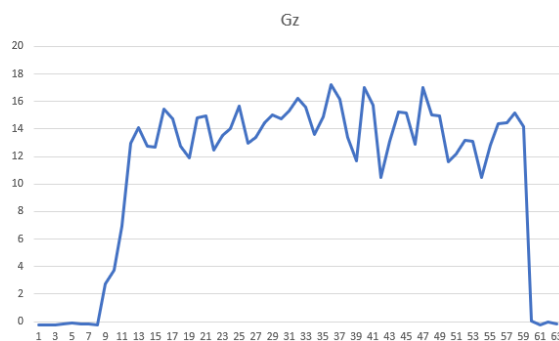


Figura 4.5: Medidas proporcionadas por el giróscopo.

Así mismo, en la figura 4.6 se reflejan los valores tomados para la componente z, respetando el signo a fin de poder discriminar no únicamente el valor de la rotación en torno al eje Z sino también el sentido de dicho giro que lo aporta el signo de la medida.

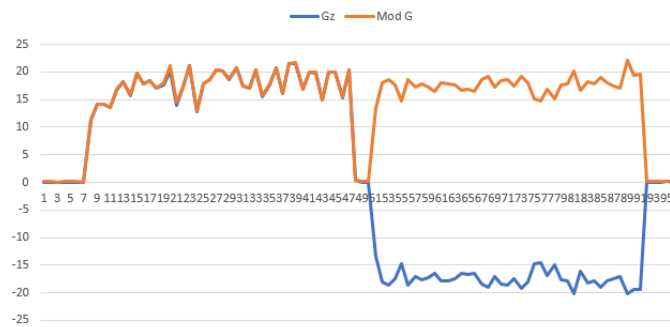


Figura 4.6: Medida del valor de la componente z del giróscopo.

Como se ha indicado anteriormente, casi toda la medida del giróscopo descansa en el eje Z y además concuerda de forma muy exacta con los valores predichos para una velocidad de giro de $14'5^{\circ}/s$, en la figura 4.5.

4.2 Caracterización del acelerómetro

En último lugar se procedió a la caracterización de acelerómetro lo cual resultó más complejo de lo esperado debido a las limitaciones técnicas del integrado MPU-9250 IMU. Además del ruido del que ya se ha hablado, la velocidad de cómputo de la plataforma de medida, no olvidar que se desea que el coste sea reducido, impide la toma de un gran número de medias en poco tiempo, 2 muestras por segundo, lo cual es suficiente para el resto de las medidas, pero escasea a la hora de medir aceleraciones en un periodo corto de tiempo. Aun con estas limitaciones se ha podido caracterizar el acelerómetro empleando únicamente tramos rectos, ver figura 4.7, para así evitar la componente de giro (y consiguiente aceleración centrípeta) y centrarse en la aceleración lineal.



Figura 4.7 Detalle del trazado para la caracterización del acelerómetro.

Con esta configuración de medida y a velocidad constante resulta obvio que la medida de aceleración proporcionada por el giróscopo debería ser nula mostrándose únicamente una componente de aceleración ante un proceso de frenado o de incremento de velocidad (sea progresivo o repentino). En la figura 4.8 se puede ver el valor del módulo del vector aceleración (en sus componentes A_x y A_y) destacándose dos máximos ligados a la fuerte aceleración inicial (en arranque) y la parada brusca final.

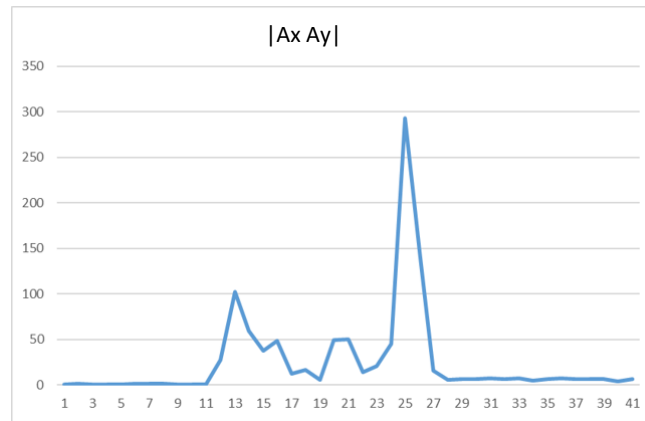


Figura 4.8: Medida del módulo de las componentes A_x y A_y .

Si se tiene en cuenta el signo de los valores medidos, se puede apreciar en la figura 4.9 cuándo se trata de una aceleración positiva, hecho que coincide con el primer máximo de la gráfica, y cuándo de un proceso de frenado (aceleración negativa o deceleración), que coincide, en este caso, con el mínimo absoluto de la gráfica.

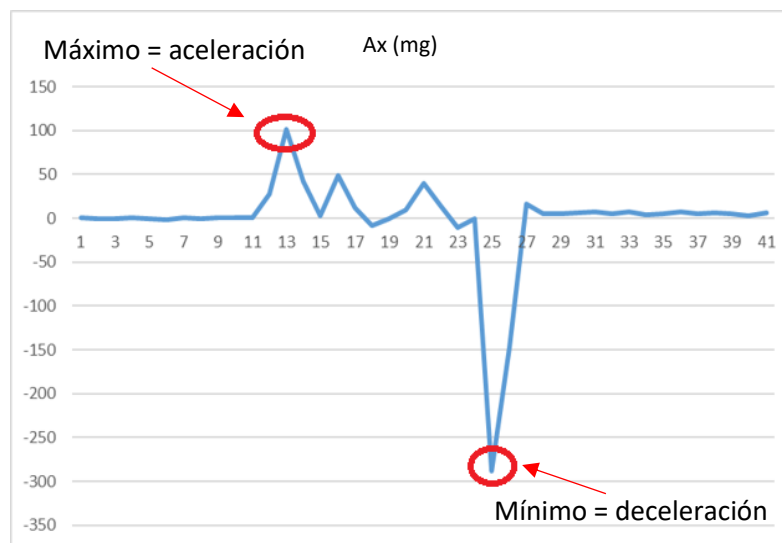


Figura 4.9: Medidas del acelerómetro teniendo en cuenta los signos.

También se puede observar algunas de las limitaciones técnicas que se han comentado anteriormente notando que los máximos y mínimos son de distinto valor (cuando a priori deberían ser idénticas para una misma aceleración), lo cual se debe a que la velocidad a la que se producen las correspondientes aceleraciones y deceleraciones es mucho mayor que la capacidad que dispone el integrado para medir y enviar los datos.

Una vez caracterizados los 6 ejes, la siguiente etapa consistió en añadir al integrado con el que se venía trabajando una placa de GPS, para ayudar a la localización en exteriores. Una de las pruebas llevada a cabo ha sido la realización de un trayecto desde la Universidad hasta un destino concreto. Una vez tomadas las medidas se procedió a analizarlas. En la figura 4.10, se muestran los valores del acelerómetro y giroscopio en función de la posición. Observar que, al comienzo del recorrido, a la derecha de la imagen, hay varias aceleraciones/deceleraciones (frenadas) y giros bruscos los cuales corresponden a las rotondas que hay en la S20.

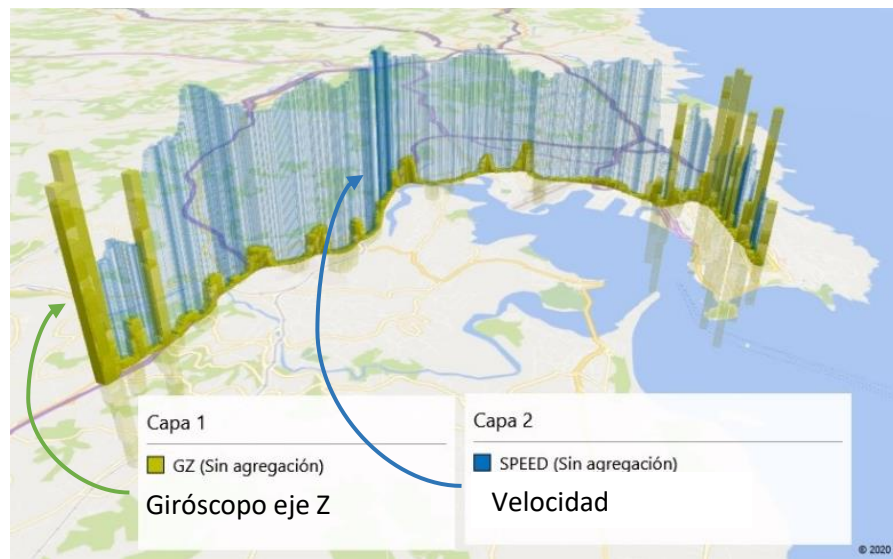


Figura 4.10 Gráfica del trayecto en coche del integrado junto con GPS.

Capítulo 5

Conclusiones y futuras líneas de trabajo

Llegados a este punto es el momento de reflexionar acerca de aquellos aspectos y consideraciones más relevantes en relación al presente trabajo final de grado. A saber,

- Se han capturado los requisitos funcionales, técnicos y de mercado de un sistema destinado a monitorizar parámetros ligados a la ergonomía de los hábitos de conducción.
- Se ha diseñado, implementado y validado un prototipo basado en un sistema embebido de la familia Arduino de bajo consumo que proporciona información en relación a los 9 ejes de los cuales 6 han sido convenientemente tratados en el trabajo (giróscopo, acelerómetro).
- Se ha concebido un escenario de validación con garantías de replicabilidad de los experimentos a fin de posibilitar la caracterización del giróscopo y acelerómetro.
- Se ha desarrollado una extensa campaña de medidas que ha posibilitado identificar técnicas destinadas a corregir derivas ligadas a los correspondientes sensores.
- Se ha testeado el sistema emplazándolo en un vehículo a fin de iniciar la caracterización de prestaciones correspondiente.

Los resultados obtenidos son más que esperanzadores en base a las medias realizadas. Nos es menos cierto que la utilización del magnetómetro, en toda su potencialidad, no ha sido todavía abordada, pero ello se subsanará con la integración de una tarjeta GPS que aportará información equivalente y a su vez permitirá consolidar los otros 6 ejes. El objetivo final es reducir al máximo los complejos sistemas de calibración que implementan algunas de las soluciones de mercado existentes. Ello abre las puertas a un enorme abanico de futuras líneas de actuación. Entre ellas sin duda la más relevante es el análisis de las prestaciones que se derivan de la integración de técnicas de procesamiento de los datos recolectados. Entre ellos cabe destacar los filtros complementarios [7], de Kalman [7] o de Kalman extendido [14] para los cuales habrá que tener en consideración el compromiso entre complejidad computacional, plataforma embebida elegida y coste de la solución.

Referencias bibliográficas

[1] CHENG, S.-H., HUANG, J.-C. & LIN, C.-J. 2012 *Proceedings - International Conference on Machine Learning and Cybernetics*. A real-time location and infant monitoring system based on active RFID, Article number 56359656, pp. 1844-1849.

[2] Vehículos e IoT, en:

<https://www.fundacionbankinter.org/ftf/tendencias/internet-de-las-cosas/ftfrefresh/impacto-iiot/vehiculos> (visitado 25-07-2020).

[3] PRAYOGO, R.C., TRIWIYATNO, A. & SUMARDI. 2018, *Proceedings - 5th International Conference on Information Technology, Computer and Electrical Engineering, ICITACEE 2018*. Quadraped Robot with Stabilization Algorithm on Uneven Floor using 6 DOF IMU based Inverse Kinematic, Article number 8576969, pp. 39-44.

[4] PITITEERAPHAB, Y., CHOITKUNNAN, P., THONGPANCE, N., KULLATHUM, K., PINTAVIROOJ, C. 2016, *3rd International Conference on Biomedical Engineering. Robot-arm control system using LEAP motion controller*, Article number 7782091, Pages 109-112.

[5] TUDOSE, L. & BUDACIU, C. 2019, *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* Volume 2019. Design and implementation of a butterfly robot system, Article number 8869463, Pages 1665-1668.

[6] VIPPALAPALLI, V., ANANTHULA, S. 2017, *International Conference on Signal Processing, Communication, Power and Embedded System*. Internet of things (IoT) based smart health care system, Article number 7955637, Pages 1229-1233.

[7] GUI, P., TANG, L. & MUKHOPADHYAY, S. 2015 *Proceedings of the 2015 10th IEEE Conference on Industrial Electronics and Applications, ICIEA 2015*. MEMS based IMU for tilting measurement: Comparison of complementary and Kalman filter based data fusion, Article number 7334442, pp. 2004-2009.

[8] HUANG, H. & ZHENG, Y.R. 2018 *Physical Communication*. 3-D localization of wireless sensor nodes using near-field magnetic-induction communications, Article number 30, pp. 97-106.

[9] CONTRERAS-RODRIGUEZ, L.A., MUNOZ-GUERRERO, R. & BARRAZA-MADRIGAL, J.A. 2017 *14th International Conference on Electrical Engineering, Computing Science and Automatic Control, CCE 2017*. Algorithm for estimating the orientation of an object in 3D space, through the optimal fusion of gyroscope and accelerometer information, Article number 8108879.

[10] YE, H., WANG, D., WU, J., YUE, Y. & ZHOU, Y. 2020, *Robotics and Computer-Integrated Manufacturing*. Forward and inverse kinematics of a 5-DOF hybrid robot for composite material machining, Volume 65, Article number 101961.

[11] LI, G. 2012. *3rd International Conference on Computing, Control and Industrial Engineering, CCIE 2012*. Quaternions and vector mapping in three-dimensional space. Volume 235, 2012, Pages 101-106.

[12] MA, L., ALBORATI, M. 2019 *IEEE 11th International Conference on Engineering Education, ICEED 2019*. Wireless Inter-Vehicle Communication among VEX Robots, Article number 8994949, pp. 78-83.

[13] Casa LGB, en: <https://www.lgb.com/> (visitado 15-02-2020).

[14] SONG, X., LI, X., TANG, W., ZHANG, W. & LI, B. 2014, *Sensors*. A hybrid positioning strategy for vehicles in a tunnel based on rfid and in-vehicle sensors, Volume 14, Issue 12, 5 December 2014, Pages 23095-23118.